Introduction

Additional introductory material on homomorphic encryption can be found on the Homomorphic Encryption Wikipedia page.

BASICS OF HOMOMORPHIC ENCRYPTION

Fully homomorphic encryption, or simply *homomorphic encryption*, refers to a class of encryption methods envisioned by Rivest, Adleman, and Dertouzos already in 1978, and first constructed by Craig Gentry in 2009. Homomorphic encryption differs from typical encryption methods in that it allows computation to be performed directly on encrypted data without requiring access to a secret key. The result of such a computation remains in encrypted form, and can at a later point be revealed by the owner of the secret key.

APPLICATIONS

Cheap cloud computing and cloud storage have fundamentally changed how businesses and individuals use and manage their data. Traditional encryption methods, such as AES, are extremely fast, and allow data to be stored conveniently in encrypted form. However, to perform even simple analytics on the encrypted data, either the cloud server needs access to the secret key, which leads to security concerns, or the owner of the data needs to download, decrypt, and operate on the data locally, which can be costly and create a logistic challenge. Homomorphic encryption can be used to simplify this scenario considerably, as the cloud can directly operate on the encrypted data, and return only the encrypted result to the

owner of the data. More complex application scenarios can involve multiple parties with private data that a third party can operate on, and return the result to one or more of the participants to be decrypted.

The yearly iDASH competition challenges the research community to push the limits and extend homomorphic encryption to new use-cases in the field of genome privacy.

SECURITY

The security of the most practical homomorphic encryption schemes is based on the *Ring-Learning With Errors (RLWE)* problem, which is a hard mathematical problem related to high-dimensional lattices. Namely, the security assumption of these encryption schemes states that if the scheme can be broken efficiently, then the RLWE problem can be solved efficiently. A long line of peer-reviewed research confirming the hardness of the RLWE problem gives us confidence that these schemes are indeed at least as secure as any standardized encryption scheme.

As was mentioned above, the RLWE problem is closely related to famous hard lattice problems which are currently considered to be secure against quantum computers. Similarly, RLWE and subsequently most homomorphic encryption schemes are considered to be secure against quantum computers, making them in fact more secure than factorization and discrete logarithm-based systems such as RSA and many forms of elliptic curve cryptography. In fact, the post-quantum cryptography standardization project organized by NIST had several submissions based on hard lattice problems similar to what modern homomorphic encryption uses.

STANDARDIZATION

There are several reasons why this is the right time to standardize homomorphic encryption.

- There is already dire need for easily available secure computation technology, and this need will be getting stronger as more companies and individuals switch to cloud storage and computing. Homomorphic encryption is already ripe for mainstream use, but the current lack of standardization is making it difficult to start using it.
- Implementations of leading schemes (CKKS, BFV, BGV, etc...) have begun to be adopted to address the needs of privacy-protected computations.
- The security properties of RLWE-based homomorphic encryption schemes can be hard to understand. The standard will present the security properties of the standardized scheme(s) in a clear and understandable form.

AVAILABILITY

Several open-source implementations of homomorphic encryption schemes exist today. Below is an incomplete list. If you would want to see your implementation being added, please contact us at contact@HomomorphicEncryption.org.

- Microsoft SEAL: A widely used open source library from Microsoft that supports the BFV and the CKKS schemes.
- PALISADE: A widely-used open source library from a consortium of DARPA-funded defense contractors that supports multiple homomorphic encryption schemes such as BGV, BFV, CKKS, TFHE and FHEW, among others, with multiparty support.
- HELib: An early and widely used library from IBM that supports the CKKS and BGV scheme and bootstrapping.
- FHEW / TFHE: Supports the TFHE scheme. (Please note that the FHEW and TFHE libraries are distinct from the FHEW and TFHE schemes which are also supported by other libraries listed on this page.)
- HeaAn: This library implements the CKKS scheme with native support for fixed point approximate arithmetic.
- Λ ○ λ (pronounced "L O L"): This is a Haskell library for ring-based lattice cryptography that supports FHE.
- NFLlib: This library is an outgrowth of the European HEAT project to explore high-performance homomorphic encryption using low-level processor

primitives.
    - HEAT: This library focuses on an API that bridges FV-NFLib and HeLIB.
    - HEAT: A HW accelerator implementation for FV-NFLlib.
- cuHE: This library explores the use of GPGPUs to accelerate homomorphic encryption.
- Lattigo: This is a lattice-based cryptographic library written in Go.
- Concrete: This library supports a custom variant of the TFHE scheme.
- EVA: A compiler and optimizer for the CKKS scheme (targeting Microsoft SEAL).